Nomes: Anderson Castilhos , Maurício Rocha,João Vicente Lessa

FASE 1

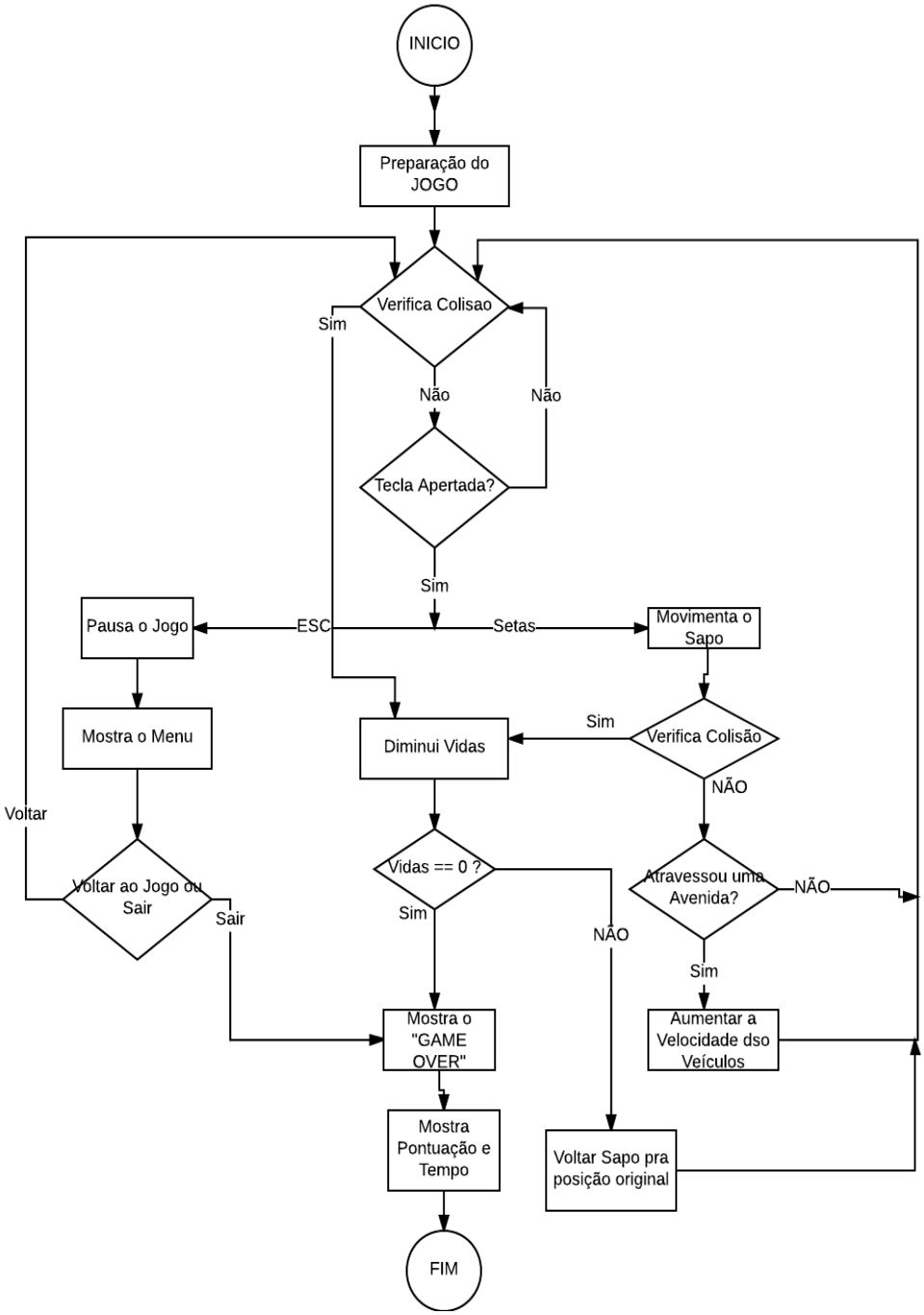
Relatório do Trabalho Prático

1ª Parte:

Primeiramente,elaboramos a pequena lista abaixo para nos ajudar a compreender melhor a mecânica do jogo descrita no enunciado e sua relação com o sistema a ser desenvolvido.

Operações:	**Realizada sobre:**	**Consequencias:**
Movimentação	O sapo	Alterar sua posição na tela
Movimentação	Os veículos	Alterar sua posição na tela
Verificar Colisões	O sapo	Se o sapo colidir com um veículo, descontar uma vida e voltar ao ponto original
Contar as Vidas	O sapo	Se as vidas chegarem a zero,mostrar mensagem de “Game Over” na tela
Calcular a Pontuação	O jogo	Mostrar a pontuação final na tela quando as vidas chegarem a zero
Aumentar a Velocidade	Os veículos	Veiculos passarao mais rapidamente

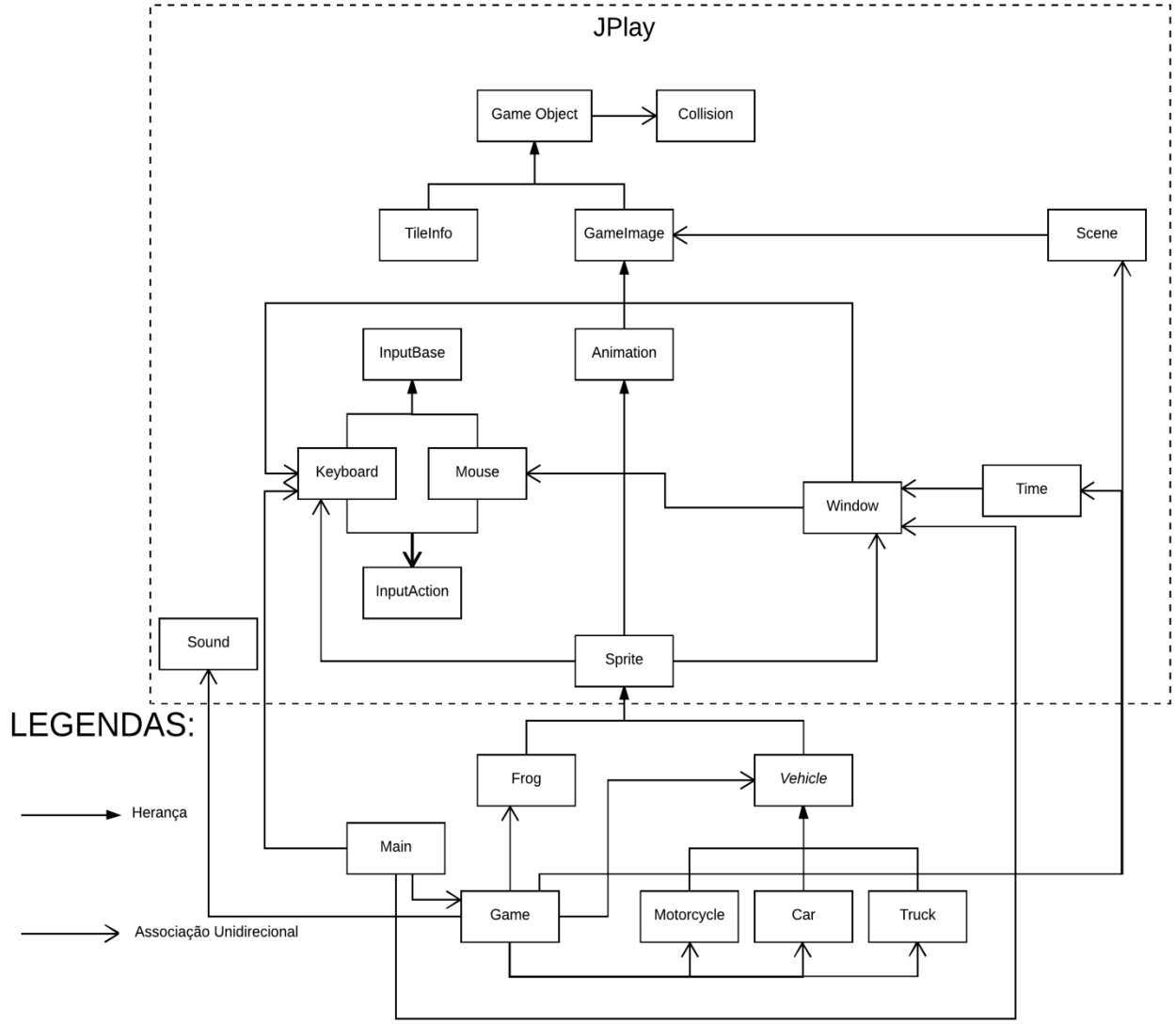
Posteriormente , criamos um fluxograma que demostra o comportamento do jogo para facilitar a identificação e o desenvolvimento das classes.



2ª Parte:

Para a futura implementação correta das classes necessárias para o bom funcionamento do jogo, nós utilizamos classes do framework Jplay para nos auxiliar,produzido pela UFF(Universidade Federal Fluminense),mais informações em http://www2.ic.uff.br/jplay/index.html e a respectiva documentação em http://www2.ic.uff.br/jplay/documentacao/index.html .

Elaboramos um diagrama de classes semelhante ao UML,porém sem mostrar os atributos e os métodos para economizar espaço.As classes dentro da área tracejada são do Jplay.



Descrição das Classes:

Package jplay

Class Summary	
Animation	Class responsible for animating a GameImage using pieces of the image, such as frames.
Collision	Class used to know whether two GameObjects collided.
GameImage	Class responsible for modeling an image.
GameObject	The most basic class presents in the framework.
InputBase	Class used to handle actions for buttons or for keys.
Keyboard	Class responsible for handling the keys of keyboard and its behavior.
Mouse	Class responsible for handling mouse actions.
Scene	Class responsible for handling a Scenario.
Sound	Class responsible for controlling the execution of sounds.
Sprite	Class responsible for controlling all actions and behaviors of sprite.
TileInfo	Class used to handle TileInfo.
Time	Class used to manipulate time.
Window	Main class of the framework.

Package Frogger:

Frog: Subclasse de Sprite responsável pelas ações e características do sapo.

Vehicle: Subclasse abstrata de Sprite responsável pelas ações e características gerais dos veículos.

Motorcycle: Subclasse de Vehicle responsável pelas ações e características específicas das motos.

Car: Subclasse de Vehicle responsável pelas ações e características específicas dos carros.

Truck: Subclasse de Vehicle responsável pelas ações e características específicas dos caminhões.

Game: Classe responsável por criar e organizar os objetos no jogo.

Main: Classe principal responsável por criar o jogo e operar o Keyboard.

Atributos e Métodos:

Frog:

private int lifes
private int direction : CIMA=1 DIRETA=2 BAIXO=3 ESQUERDA=4
private boolean moving

public Frog(int x, int y): Construtor que seta as coordenadas iniciais do sapo
public void avoidWalls(Scene scene): Método que trata da colisão do sapo com o cenário estático e com as bordas da janela.
public boolean vehicleColision(Vehicle vehicle):Retorna true se houve colisão do sapo com o veículo,caso contrário retorna false.
public void move(Keyboard keyboard):Método que trata da movimentação do sapo
private boolean tileColision(GameObject obj,TileInfo tile):Auxiliar do avoidWalls.

private boolean verticalColision(GameObject obj,GameObject vehicle): Auxiliar do avoidWalls.

private boolean horizontalColision(GameObject obj,GameObject vehicle): Auxiliar do avoidWalls.

public void toStartPosition(): Manda o sapo de volta para a posição inicial.

Vehicle:

protected double speed

public Vehicle(String fileName,int numFrames):Construtor.
public Vehicle(int x,int y,String fileLocation):Construtor.
public void move():Faz o veículo se movimentar.
public void accelerate():Aumenta a velocidade do veículo.

Motorcycle:

public Motorcycle(): Construtor.
public Motorcycle(int x,int y): Contrutor.
Ambos definem a imagem de sua animação e sua velocidade inicial

Car:

public Car (): Construtor.
public Car (int x,int y): Contrutor.
Ambos definem a imagem de sua animação e sua velocidade inicial

Truck:

public Truck (): Construtor.
public Truck (int x,int y): Contrutor.
Ambos definem a imagem de sua animação e sua velocidade inicial

Main:

public static void main(String[] args): Método principal

Game:

private Window window;
public Scene scene;
private Frog frog;
private Motorcycle moto1[];
private Motorcycle moto2[];
private Car car1[];
private Car car2[];
private Truck truck1[];
private Truck truck2[];
Além de possuir todas as constantes do jogo

public Game(Window window): Construtor que inicializa todos os objetos do jogo.
public void play(Keyboard keyboard):Anima e faz a interação entre os objetos do jogo
public void addVehicles(Vehicle vehicle[],int y,int numVehicles):Adiciona os veículos no jogo.
public void animateVehicles(Vehicle vehicle[],int numVehicles):Anima os veículos no jogo.
public void initializeVehicles(Motorcycle moto[],Car car[],Truck truck[],int numMotos,int numCars,int numTrucks): Inicializa os veículos no jogo.

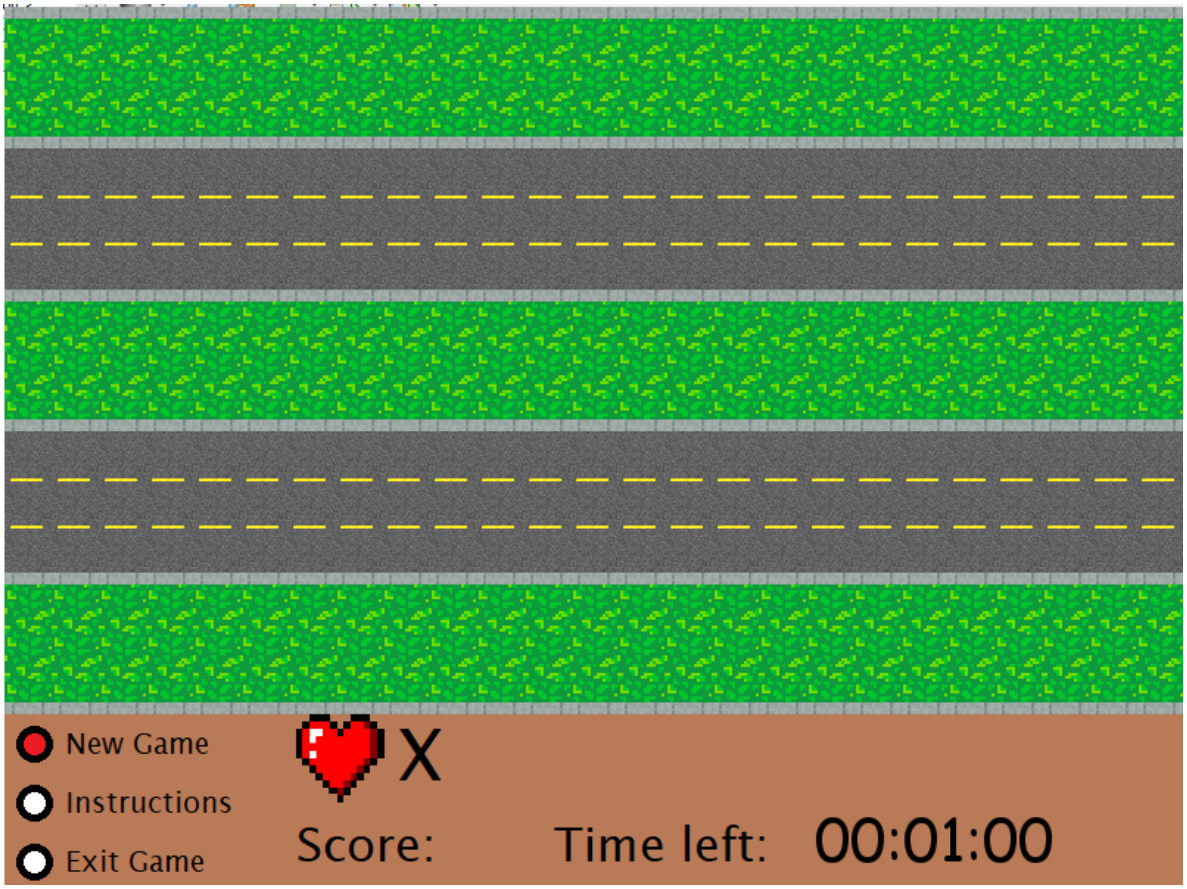
Os métodos abaixo fazem parte da classe game e serão desenvolvidas com o intuito de diminuir a quantidade de linhas de código do método play.

public void accelerateVehicles(Vehicle vehicle1[],Vehicle vehicle2[],Vehicle vehicle3[],int numVehicle1,int numVehicle2,int numVehicle3):Acelera os veículos.

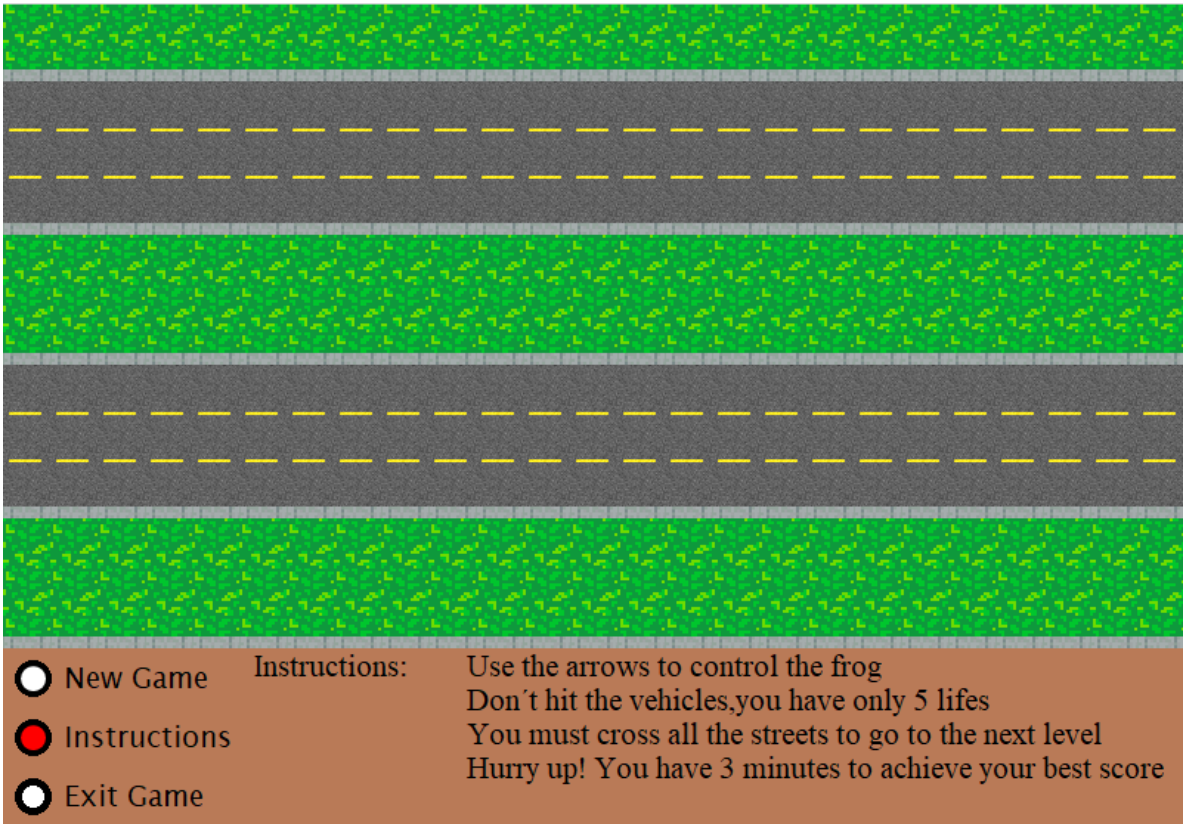
public boolean frogVehicleColision(Frog frog,Motorcycle moto[],Car car[],Truck truck[],int numMotos,int numCars,int numTrucks):Verifica a colisão do sapo com todos os veículos do jogo.

3ª Parte:

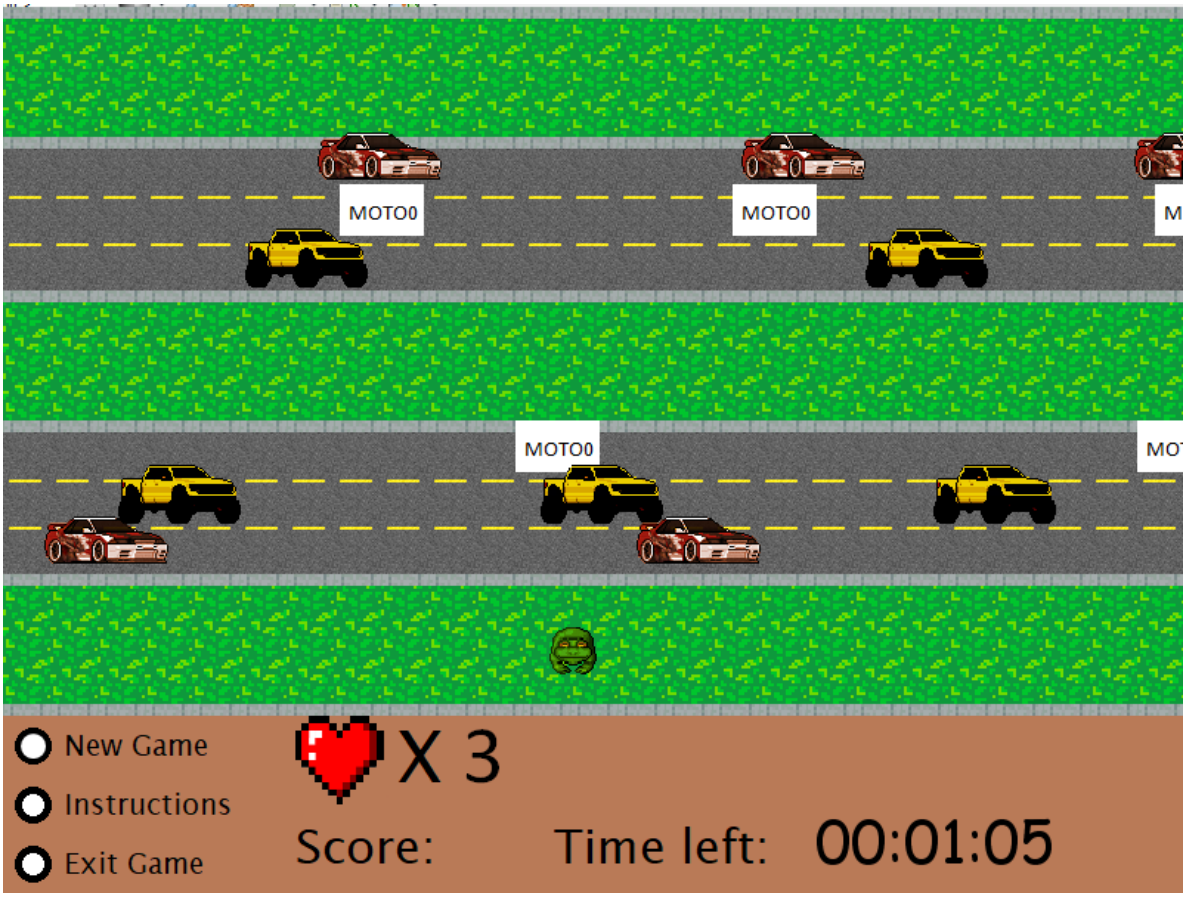
A nossa interface com o usuário consistirá em uma janela 800x600,que é iniciada printando o cenário do jogo(presente na maior parte da tela) e o menu do jogo(presente em uma pequena área na parte inferior da janela).Os screenshots ainda são “protótipos” e estão sujeitos a aprimoramentos gráficos.



O usuário então pode escolher se quer começar o jogo ou se quer ler as instruções.



Ao escolher a opção de começar o jogo, o jogo é iniciado e o sapo e os veículos aparecem na tela.(imagem da moto ainda pendente)



Ao apertar durante o jogo ESC ,o jogo é pausado e o usuário pode visualizar seu tempo restante e sua atual pontuação na área do menu,além de poder escolher entre as opções do menu.